Mécanismes d'abstraction dans une représentation de le connaissance centrée objet (R.C.O.)

Sophie Le Peutrec, Sophie Robin

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Mécanismes d'abstraction dans une représentation de la connaissance centrée objet (R.C.O.)

Stéphane LE PEUTREC
Sophie ROBIN

N° 1840
Janvier 1993

PROGRAMME 3

Intelligence artificielle,
Systèmes cognitifs et
Interaction homme-machine

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX FRANCE
Tél. : 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

Stéphane Le Peutrec, Sophie Robin

janvier 1993
Publication Interne n° 694 - 40 pages - Programme 3

Mécanismes d'abstraction dans une représentation de la connaissance centrée objet (R.C.O.).

Les travaux réalisés sur l'abstraction se situent soit dans le cadre de la démonstration, soit dans le cadre de l'apprentissage. L'abstraction est alors utilisée pour transformer un problème complexe en un problème simplifié dans le but d'alléger sa résolution.Or le système GIDE a mis en évidence le besoin d'intégrer des mécanismes d'abstraction aux R.C.O. pour permettre certains types de raisonnements. Dans cet article, nous présentons, tout d'abord, brièvement les travaux sur l'abstraction dans le cadre de la résolution de problèmes. Puis, nous en montrons les similitudes avec les différentes formes d'abstraction que nous avons retenues au sein d'une R.C.O. Ensuite, nous développons une formalisation de l'abstraction plus adaptée au cadre des R.C.O. et en particulier à l'héritage multiple, dans laquelle une abstraction est représentée par une règle de réécriture associée à un contexte sémantique.

mots clés: abstraction, représentation centrée objet, systeme de réécriture.

Abstraction mechanisms for an object oriented representation (O.O.R.).

Works on abstraction concern either resolution problems or machine learning. In all these cases, abstraction is used to convert a complex problem A to a simpler one B, the solutions of B guide in searching for the solutions of A. But the GIDE system shows the necessity to integrate abstraction into O.O.R. to allow some particular reasoning. In this paper, we briefly present works on abstraction dealing with the resolution of problems. Then, we show the similarity between these works and the different types of abstraction we have dicerned in O.O.R. Afterwards, we propose a formalism for abstraction more adapted to O.O.R. and in particular to the multiple inheritance. In this formalism, an abstraction is represented by a rewriting rule.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE (U.R.A. 227) UNIVERSITE DE RENNES I INSA DE RENNES
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE (UNITE DE RECHERCHE DE RENNES)

Table des matières

1 Introduction **4**
- 1.1 Cadre du projet : le système GIDE 4
- 1.2 Plan du document 5

2 Abstraction **6**
- 2.1 Introduction 6
- 2.2 Les travaux de A.Giordana 6

3 L'abstraction dans les R.C.O. **8**
- 3.1 Analyse du problème 8
- 3.2 Similitudes avec les travaux de Giordana 10
- 3.3 Discussion 12

4 Comment formaliser l'abstraction dans les R.C.O. ? **14**
- 4.1 Introduction 14
  - 4.1.1 But du présent chapitre 14
  - 4.1.2 Conventions de notation 14
  - 4.1.3 Plan du chapitre 15
- 4.2 Représentation des instances 15
  - 4.2.1 Représentation des instances dans le cas de l'héritage simple 15
  - 4.2.2 Représentation des instances dans le cas de l'héritage multiple 17
- 4.3 Schéma général d'une abstraction 18
- 4.4 L'abstraction dans le cas de l'héritage simple 19
  - 4.4.1 Les abstractions par généralisation (g-abstractions) 19
  - 4.4.2 Les abstractions par synthèse (s-abstractions) 22
- 4.5 L'abstraction dans le cas de l'héritage multiple 24
  - 4.5.1 La projection d'attributs 24
  - 4.5.2 La généralisation de classes 24
- 4.6 Les abstractions composées 25
  - 4.6.1 Deux types d'abstraction simples 25
  - 4.6.2 Les g-abstractions composées 25
  - 4.6.3 Les s-abstractions composées 27
  - 4.6.4 Les g/s-abstractions composées 27

5 Conclusion **28**
5.1 Les acquis 28
5.2 Les perspectives 29

A Figures **32**

Chapitre 1

Introduction

1.1 Cadre du projet : le système GIDE

Le système GIDE a pour but la gestion de dossiers médicaux de patients épileptiques [10] [8]. La durée de la maladie (plusieurs années), ainsi que les nombreux facteurs qui s'y rattachent, nécessitent la prise en compte de données volumineuses et très diverses.

Les besoins pour la gestion de tels dossiers sont:

- la modularité des connaissances,
- la description des liens entre les nombreux types d'information,
- la personnalisation des dossiers.

Le système doit aider le médecin dans la détermination de diagnostics et de thérapeutiques, d'où la nécessité d'implanter d'importants traitements de données.

Ce sont ces différentes raisons qui ont fait préférer un modèle de représentation des connaissances du type Représentation Centrée Objets (R.C.O.) [1].

L'une des fonctionnalités principales du système GIDE est de permettre la synthèse d'un dossier médical, notamment en début de consultation. Les données doivent donc évoluer avec le temps. Par exemple, les dernières crises du patient peuvent être à l'heure actuelle décrites précisément, et résumées dans plusieurs années en fréquence mensuelle, voire annuelle.

Les différentes fonctions à mettre en œuvre n'utilisent pas forcément toutes les mêmes informations et ne nécessitent pas toutes le même grain de précision.

Exemple:

- Considérons une fonction telle que:
  "de tous les traitements suivis par le malade, ne ressortir que les traitements pertinents c'est à dire ceux qui ont eu le meilleur effet ou ceux que le malade n'a pas tolérés par exemple."
  Une telle fonction va s'intéresser aux fréquences des crises pendant les périodes de suivi des différents traitements tout en leur associant les effets secondaires éventuels intervenus ou non (chutes des cheveux, prise de poids,...).

- Soit une seconde fonction qui s'attache à
  "afficher la courbe de fréquence des crises pour faire état de l'évolution de la maladie, en faisant ressortir les périodes sur lesquelles la fréquence subit des variations qui pourraient être liées à des paramètres tels que le changement de traitement, les activités socio-professionnelles, ..."
  Cette fonction prend en compte les activités du patient, ce qui n'est pas le cas de la précédente. De plus, ces deux fonctions ne vont pas utiliser le même grain de précision pour les traitements, en particulier au niveau des effets secondaires.

Ces exemples montrent que la synthèse de dossier nécessite de condenser, voire d'ignorer certaines informations suivant le contexte d'utilisation du dossier. De tels mécanismes de raisonnement ont été regroupés sous le terme "d'abstraction".

Une première étude faite par M.Courant et S.Robin sur l'abstraction dans le système GIDE a permis de dégager un certain nombre de mécanismes d'abstraction suffisamment généraux pour sortir du cadre de GIDE[8].

Dans le présent document, il s'agit de compléter cette analyse à la lumière d'une étude bibliographique sur les travaux liés à l'abstraction. Puis nous proposons une formalisation homogène des mécanismes d'abstraction retenus.

1.2 Plan du document

Le document comporte quatre chapitres. Le chapitre 2 fait le point sur les différents travaux réalisés autour de l'abstraction.

Le chapitre 3 répertorie les formes d'abstraction que nous avons retenues au sein d'une R.C.O. et en montre les similitudes avec les travaux réalisés sur l'abstraction, présentés dans le deuxième chapitre.

Le chapitre 4 développe une formalisation de l'abstraction au sein d'une R.C.O.

Enfin, le chapitre 5, qui tient lieu de conclusion, présente les points forts et les limites de ce travail.

Chapitre 2

Abstraction

2.1 Introduction

Si l'on se réfère à la définition fournie par le dictionnaire, l'abstraction est une opération de l'esprit qui isole d'une notion un élément en négligeant les autres.

Exemples:

- dans l'ensemble des poissons, ne considérer que les poissons de mer,
- parmi les participants d'une compétition sportive, ne s'intéresser qu'aux trois premiers.

Les mécanismes de raisonnement utilisés dans GIDE[10] [8] et énoncés en introduction telles que

- exhiber les traitements les plus efficaces,
- résumer l'ensembles des crises par une fréquence,

sont donc typiquement des abstractions. De tels mécanismes semblent suffisamment généraux pour sortir du cadre de GIDE. Pourtant, le problème de l'abstraction dans les R.C.O. n'a pas été jusqu'ici abordé en tant que tel. Les travaux réalisés sur l'abstraction se situent soit dans le cadre de la démonstration, soit dans le cadre de l'apprentissage. Parmi les premiers travaux sur l'abstraction, ceux de D.Plaisted [6] ont servi de base à de nombreux autres travaux [2],[4],[9],[5],[7]. Nous présentons ici les travaux réalisés par A.Giordana[4]. Nous mettons en évidence leurs similitudes avec les besoins rencontrés dans GIDE dans le chapitre 3.

2.2 Les travaux de A.Giordana

En s'appuyant sur les travaux de Plaisted et de Tenenberg, Giordana élargit la notion d'abstraction [4]. Il définit deux classes d'abstraction :

- **la généralisation** est une abstraction au sens de Plaisted et Tenenberg. La généralisation produit, à partir de formules du 1^{er} ordre, de nouvelles formules du 1^{er} ordre en supprimant les détails (perte d'informations).
  Exemple: ne garder que le 1^{er} chiffre d'un résultat de prise de tension.

- **la synthèse** est mise en évidence par Giordana. La synthèse produit, à partir de formules du 1^{er} ordre, de nouvelles formules du 1^{er} ordre en synthétisant l'information initiale à l'aide de fonctions (synthèse d'informations).
  Exemple: considérer le résultat d'une prise de tension comme anormal ou normal.

Giordana donne une nouvelle définition de l'abstraction qui permet de réunir ces deux formes d'abstraction.

Définition: *Une théorie d'abstraction TA d'un langage L vers un langage L' est un ensemble consistant de définitions de ce type:*
$\forall \bar{x}, \bar{y}\ [(y1 = f1(\bar{x}), ..., yn = fn(\bar{x})) \rightarrow (\varphi(\bar{x}) \leftrightarrow \psi(\bar{y}))]$
où φ *est une formule appartenant à L,*
ψ *est une formule appartenant à L',*
$\bar{x}$,$\bar{y}$ *sont respectivement les variables libres de* φ *et* ψ.

Grâce aux relations $y1 = f1(\bar{x}), ..., yn = fn(\bar{x})$, chaque yi est le résultat d'une application sur $\bar{x}$. Ce sont les fonctions fi qui permettent la synthèse. La base abstraite obtenue n'est composée que par des formules ψ de L'. Les formules initiales ne pouvant être abstraites sont oubliées.

Dans [4], Giordana présente différentes formes d'abstraction. Nous en donnons ici quelques exemples.

- **élimination d'un groupe de prédicats.**
  Cette forme d'abstraction est la même que celle utilisée par Knoblock dans [7].
  Soit la base initiale: $P1, ...Pn, Q1, ..., Qm$
  $TA = \{\forall \bar{x}[P1 \leftrightarrow P1], \forall \bar{x}[P2 \leftrightarrow P2], ..., \forall \bar{x}[Pn \leftrightarrow Pn]\}$
  Cette théorie d'abstraction élimine les prédicats $Q1, ..., Qm$.
  Elle est consistante ssi il existe au moins un modèle de la base initiale qui ne contient aucun prédicat Qi.

- **abstraction sur les prédicats** (attributs inchangés),
  Les abstractions sont du type de celles décrites par Tenenberg.
  $\forall \bar{x}[P1(\bar{x}) \vee P2(\bar{x}) \vee ... \vee Pn(\bar{x}) \leftrightarrow Q(\bar{x})]$,
  avec le cas particulier de $\forall \bar{x}[P1(\bar{x}) \wedge ... \wedge Pn(\bar{x}) \leftrightarrow Q(\bar{x})]$ qui est particulièrement utile pour construire des raisonnements inductifs, dans le cadre de l'apprentissage par exemple.

- **abstraction sur les attributs** (grâce aux fonctions fi).
  $\forall \bar{x}, y[(y = f(\bar{x})) \rightarrow (P(\bar{x}) \leftrightarrow Q(y))]$.

Chapitre 3

L'abstraction dans les R.C.O.

Dans ce chapitre, nous étudions, tout d'abord, les mécanismes d'abstraction exhibés dans le cadre du système GIDE. Chacun des mécanismes retenus est illustré par un exemple emprunté au système GIDE. Puis, nous cherchons à rattacher cette étude aux travaux réalisés sur l'abstraction et présentés au chapitre 2 en montrant leurs similitudes puis leurs divergences.

3.1 Analyse du problème

L'étude effectuée sur les mécanismes d'abstraction utilisés dans GIDE [8] avait permis de dégager deux classes d'abstraction : la généralisation et la synthèse, ce qui rejoint l'analyse faite par Giordana (cf 2.2) [4]. Nous avons donc scindé les formes d'abstraction rencontrées dans le cadre des R.C.O., en deux groupes :

- d'une part, celles qui correspondent à un raisonnement de type généralisation (perte d'information),
- d'autre part, celles qui correspondent à un raisonnement de type synthèse (condensé d'information).

Dans le premier groupe, on peut relever les formes d'abstractions élémentaires suivantes :

- la *projection d'attributs*, qui permet de ne garder que les attributs porteurs d'information dans un contexte donné,
- la *généralisation de classe*, qui permet d'oublier la classe d'instanciation d'une instance ainsi que ses attributs propres,
- la *généralisation par élément significatif*, qui réduit un ensemble d'instances à un élément significatif de cet ensemble.

Dans le second groupe, nous distinguons les formes suivantes :

- la *synthèse simple*, qui synthétise les informations d'une instance en une instance d'une autre classe ou en une valeur littérale,
- la *synthèse globale*, qui synthétise les informations d'un ensemble d'instances d'une classe en une instance d'une autre classe ou en une valeur littérale.

Les exemples qui suivent permettent d'illustrer chacune des formes d'abstraction retenues. Ces exemples seront repris dans le paragraphe 3.2 qui s'attache à montrer les liens entre nos travaux et les travaux réalisés sur l'abstraction. Toutes les figures citées ont été regroupées en annexe A.

Exemple 1 : *projection d'attributs.*

Supposons que l' **on désire connaître les dates des crises d'un patient**. Ceci revient à ne conserver que l'information (l'attribut) *date* pour une instance de la classe *crise*, et à ignorer les autres attributs.
On a (fig A.1a) et on veut (fig A.1b).

Exemple 2 : *généralisation de classes.*

Imaginons qu'une analyse représente soit une prise de sang soit une prise de tension et que l'**on souhaite connaître toutes les analyses** effectuées pour un patient donné. Au sein d'une R.C.O., cet exemple revient à considérer les instances des classes *prise-de-tension* et *prise-de-sang* comme instances de la classe *analyse* en ne conservant que les attributs de *prise-de-tension* et de *prise-de-sang* hérités de la classe *analyse* .
On a (fig A.2a) et on veut (fig A.2b).

Exemple 3 : *généralisation par un élément significatif.*

On souhaite abstraire l'ensemble des prises de tension d'un patient donné en ne retenant que la prise de tension la plus récente. Dans les R.C.O., cette forme de raisonnement revient à isoler une instance de la classe *prise-de-tension* en la comparant aux autres instances de la même classe.
On a (fig A.3a) et on veut (fig A.3b).

Exemple 4 : *synthèse simple.*

On veut abstraire le résultat d'une prise de tension par un bilan d'examen. Ce bilan peut-être représenté par une valeur littérale telle que "normal","hypertension" ou"hypotension". Imaginons que la classe *prise-de-tension* possède 3 attributs : *premier-chiffre, deuxième-chiffre* et *date*. Pour une instance de *prise-de-tension*, on veut réaliser une synthèse d'informations.
On a (fig A.4a) et on veut (fig A.4b).

Exemple 5 : *synthèse globale.*

On veut abstraire l'ensemble des prises de sang d'un patient par la moyenne du taux de cholestérol. Au sein d'une R.C.O., cet exemple revient à abstraire les instances de la classe *prise-de-sang* par une valeur littérale qui correspond à la moyenne du taux de cholestérol.
on a (fig A.5a) et on veut (fig A.5b).

3.2 Similitudes avec les travaux de Giordana

Nous venons d'exhiber 5 formes d'abstraction élémentaires utilisées dans le cadre des R.C.O. Dans ce paragraphe, nous cherchons à montrer que les travaux menés sur l'abstraction au sein d'une R.C.O. rejoignent tout à fait, les travaux menés sur l'abstraction dans le cadre plus général de la résolution de problèmes.

Nous reprenons un à un les exemples 1 à 5 présentés en 3.1 en montrant comment chacun d'eux pourait être modélisé à l'aide du formalisme proposé par Giordana.

Exemple 1 : *projection d'attributs.*

- Comme il est précisé au paragraphe 2.2, Giordana s'est basé sur les travaux de Plaisted et les a complétés, dans le but d'offrir une plus grande variété de formes d'abstraction. Ainsi le mécanisme d'abstraction mis en œuvre dans cet exemple pourrait être modélisé par Giordana grâce à la théorie d'abstraction suivante :

  $$TA1 = \{[crise(date, ...) \leftrightarrow crise(date)]\},$$

  où $TA1$ est une théorie d'abstraction par généralisation (cf 2.2). On avait déjà noté auparavant, que toute les formes d'abstraction traitées par Plaisted étaient regroupées par Giordana sous le terme d'abstraction par généralisation.

Exemple 2 : *généralisation de classes.*

- Avec le formalisme de Giordana, on pourrait utiliser la théorie d'abstraction suivante :

$$TA2 = \{\forall x, y, z, w[(\textit{prise-de-tension}(x, y, z) \vee \textit{prise-de-sang}(x, y, w)) \leftrightarrow \text{analyse}(x, y)]\}$$

où $TA2$ est une théorie d'abstraction par généralisation.

Exemple 3 : *généralisation par un élément significatif.*

- Contrairement au formalisme de Plaisted, le formalisme de Giordana permet une telle utilisation de l'information. Pour isoler la *prise-de-tension* la plus récente on peut utiliser la théorie d'abstraction suivante :

$$TA3 = \{\forall x1, y1, z1, ..., xn, yn, zn[A1 = f(x1, ., zn), A2 = f(x1, ., zn), A3 = f(x1, ., zn) \rightarrow (\textit{prise-de-tension}(x1, y1, z1) \wedge ... \wedge \textit{prise-de-tension}(xn, yn, zn) \leftrightarrow \textit{prise-de-tension}(A1, A2, A3))]\}$$

avec $f : x1, ..., zn \mapsto A3 = zi = max(z1, ..., zn), A2 = yi, A1 = xi$,
où $TA3$ est une théorie d'abstraction par synthèse (cf 2.2).

Exemple 4 : *synthèse simple.*

- Le formalisme de Giordana permet la synthèse de l'information. Pour l'exemple traité, on peut utiliser une théorie d'abstraction de la forme :
$TA4 = \{\forall \bar{x}, y[y = f(\bar{x}) \rightarrow (\textit{prise-de-tension}(\bar{x}) \leftrightarrow \textit{bilan-prise-de-tension}(y))]\}$
avec $f : x1, x2, x3 \rightarrow$ si $x1 \geq 15$ alors y=hypertension
si $x1 \leq 10$ alors y=hypotension
sinon y=normal,
où $TA4$ est une théorie d'abstraction par synthèse .

Exemple 5 : *synthèse globale.*

- Les travaux de Giordana modélisent cet exemple grâce à la théorie d'abstraction suivante :

$$TA5 = \{\forall \bar{x_1}, ..., \bar{x_n} [\textit{moyenne} = f(\bar{x_1}, ..., \bar{x_n}) \rightarrow (\textit{prise-de-sang}(\bar{x_1}) \wedge ... \wedge \textit{prise-de-sang}(\bar{x_n})) \leftrightarrow \textit{moyenne}]\}$$

où $TA5$ est une théorie d'abstraction par synthèse, et *moyenne* est une variable instanciée par le résultat de l'application f sur les arguments $\bar{x_1}, ..., \bar{x_n}$.

3.3 Discussion

Dans ce chapitre, nous avons présenté différentes formes d'abstraction dans le cadre des R.C.O. que l'on peut regrouper en deux grandes "catégories" : **abstractions par généralisation** , qui correspondent à un oubli d'information, et les **abstractions par synthèse**, qui correspondent à un condensé d'information.

Le paragraphe précédent a permis de montrer que nos travaux sont fortement connectés à des travaux plus généraux sur l'abstraction dans le cadre de la résolution de problèmes. Toutefois, on peut dès maintenant noter qu'il y a divergence sur deux principaux points : d'une part sur la signification de la notion de synthèse d'information, et d'autre part sur la restriction d'un ensemble à un de ses sous-ensembles.

• Regardons plus attentivement, le cas de l'abstraction *généralisation par élément significatif* qui se trouve illustrée par l'exemple 3. Cette forme d'abstraction se trouve rangée dans la catégorie synthèse par Giordana. En effet, pour Giordana la distinction entre généralisation et synthèse ne repose pas uniquement sur le fait qu'il s'agit d'oublier de l'information ou de condenser l'information. La distinction se fait plutôt sur le fait que les abstractions de type synthèse nécessitent d'introduire une fonction qui permet le calcul de l'information abstraite à partir de l'information de départ.

Pour ce qui nous concerne, la distinction se fait réellement sur l'oubli d'information, pour la généralisation, et le condensé d'information, pour la synthèse. Dans le cadre des R.C.O., on peut alors remarquer qu'une généralisation correspond à une abstraction dite **sans changement de concept**, qui transforme une instance ou un ensemble d'instances d'un concept donné en une instance du même concept ou du concept père. Par contre, la synthèse correspond à une abstraction dite **avec changement de concept**, qui transforme une instance ou un ensemble d'instances d'un concept donné en une instance d'un autre concept de base issu d'un chemin d'héritage différent.

• Considérons l'exemple suivant :

"Parmi l'ensemble des prises de tension, ne considérer que celles dont le premier chiffre est supérieur à 15". Ce problème, représenté dans une hiérarchie de classes, revient à déterminer un sous-ensemble de l'ensemble global des instances de la classe *prise-de-tension* en rattachant chacune de ces instances à une sous-classe, éventuellement virtuelle, de la classe *prise-de-tension* (cf fig A.6).

Nous exerçons donc une spécialisation sur l'ensemble d'instances initial. Ainsi, au sein d'une R.C.O., cet exemple n'est pas considéré comme une abstraction, puisque la spécialisation est le processus inverse de la généralisation, et qu'elle ne correspond pas non plus à une synthèse d'information, telle que nous l'avons défini en 3.1.

Par contre, dans les travaux de Giordana, cet exemple est modélisable par une théorie d'abstraction de la forme :

$$TA7 \;=\; \{\forall \bar{x}\, [vrai = f(\bar{x}) \;\rightarrow\; (\textit{prise-de-tension}(\bar{x}) \;\leftrightarrow\; \textit{prise-de-tension}(\bar{x}))]\}$$

où $TA7$ est une théorie d'abstraction par synthèse. Chacune des *prise-de-tension* ne satisfaisant pas la relation $vrai = f(\bar{x})$ est éliminée de la base initiale.

Ainsi, bien que la restriction d'un ensemble par un de ses sous-ensembles dans la logique du 1er ordre soit un cas d'abstraction, nous ne la considérons pas comme telle au sein d'une R.C.O.

Chapitre 4

Comment formaliser l'abstraction dans les R.C.O.?

4.1 Introduction

4.1.1 But du présent chapitre

Nous venons de montrer les similitudes entre les travaux réalisés sur l'abstraction et les besoins rencontrés dans les R.C.O. Notre but est d'intégrer des mécanismes d'abstraction au sein d'une R.C.O. Les formalismes qui ont été présentés au chapitre 2 pour modéliser l'abstraction sont adaptés à la logique du premier ordre. Il s'agit pour nous maintenant, en s'inspirant de ces formalismes, de proposer une formalisation des mécanismes d'abstraction plus adaptée aux particularités des R.C.O. et notamment au mécanisme d'héritage multiple.

L'abstraction est un mécanisme qui transforme la connaissance de base que l'on a d'un domaine donné, pour l'utiliser dans le cadre de raisonnements particuliers (cf exemples du chapitre 3). La connaissance de base est composée d'instances. L'abstraction doit donc manipuler des instances. Dans ce chapitre, nous proposons un formalisme de représentation des instances et des ensembles d'instances qui permet ensuite de modéliser aisément les différentes formes d'abstraction.

4.1.2 Conventions de notation

Pour faciliter la compréhension des exemples à venir et afin d'uniformiser la présentation, nous associons à chaque type d'objet rencontré dans une R.C.O., une figure particulière. Ainsi, une classe est représentée par un ovale, la classe racine de la hiérarchie, dont toutes les classes sont descendantes est représentée par un triangle et modélisée par le symbole C_0, tandis qu'une instance est représentée par un rectangle.

Exemple:

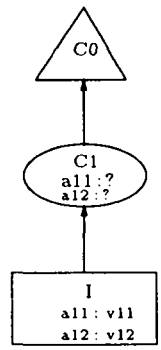


Figure 4.1 : *notation*

4.1.3 Plan du chapitre

Avant de proposer une formalisation des différentes formes d'abstraction, il est nécessaire de définir un formalisme de représentation des instances, qui sont les objets effectivement manipulés par l'abstraction.

Notre but est de modéliser de façon homogène les différentes formes d'abstraction, introduites dans le chapitre 3, au sein d'une R.C.O. Aussi, nous proposons une définition générique de l'abstraction au sein d'une R.C.O. Afin de faciliter la compréhension de la modélisation choisie pour représenter les différentes formes d'abstraction au sein d'une R.C.O., nous présentons tout d'abord, la formalisation retenue dans le cadre de l'héritage simple. Puis, nous étendons les notions introduites au cas de l'héritage multiple. Ensuite, nous développons les compositions d'abstractions.

4.2 Représentation des instances

Nous proposons, tout d'abord, un formalisme de représentation des instances dans le cadre de l'héritage simple. Puis, nous l'étendons au cas de l'héritage multiple.

4.2.1 Représentation des instances dans le cas de l'héritage simple

Dans le cadre de l'héritage simple, pour toute instance I, appelée instance simple, il n'existe qu'un seul chemin d'héritage allant de cette instance jusqu'à la racine C_0 de la hiérarchie. La connaissance de la classe d'instanciation de I et des valeurs de ses attributs suffit alors à définir l'instance I. Toutefois, pour faciliter l'extension du formalisme au cas de l'héritage multiple, la représentation proposée dans le cas de l'héritage simple fait également

apparaître le chemin d'héritage de I en rattachant, d'autre part, chacun des attributs à la classe qui les a déclarés.

Définition 1 : *Soient n classes* $C_1, ..., C_n$ *telles que :*

- *pour tout* $i \in \{1, ..., n-1\}, C_{i+1}$ *est une spécialisation de* C_i,
- *les instances de la classe* C_n *sont définies par les attributs des classes* $C_1, ..., C_{n-1}$ *et des attributs propres de* C_n. *Les attributs propres à la classe* C_i *sont notés* $A_i^1, ..., A_i^{p_i}$.

Soit I une instance simple de la classe C_n. *Elle peut être dénotée par le terme :*
$C_n(v_n^1, ..., v_n^{p_n} : C_{n-1}(v_{n-1}^1, ..., v_{n-1}^{p_{n-1}} : ... : C_1(v_1^1, ..., v_1^{p_1} : C_0)...))$
noté en abrégé $C_n(v_n^1, ..., v_n^{p_n}, ..., v_1^1, ..., v_1^{p_1})$, *où* v_i^j *est la valeur associée à l'attribut* A_i^j.

Exemple :

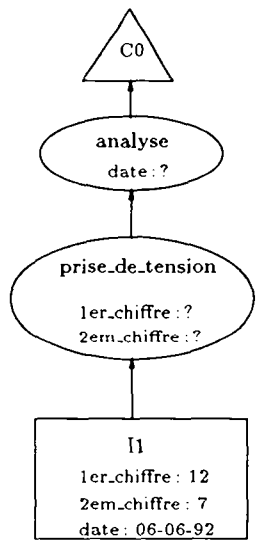


Figure 4.2 : *héritage simple*

l'instance I1 est représentée par le terme :
prise-de-tension(12,7 : *analyse*(06-06-92 : C_0))
noté en abrégé *prise-de-tension*(12,7,06-06-92)

Remarque :
La classe *analyse* est une surclasse de la classe *prise-de-tension*, donc I1 est aussi une instance d'*analyse*. Les attributs 1^{er}-*chiffre* et 2^{em}-*chiffre* ne sont déclarés que dans la classe *prise-de-tension*. Si on veut considérer I1 comme une instance d'*analyse*, il faut ignorer les valeurs des attributs 1^{er}-*chiffre* et 2^{em}-*chiffre* ainsi que l'appartenance de I1 à la classe *prise-de-tension*. Elle sera alors représentée par le terme *analyse*(06-06-92).

Le formalisme choisi permet aussi de représenter des classes. Dans ce cas, on représente par une variable tout attribut non instancié par la classe.

Exemple :
Si on reprend l'exemple illustré par la figure 4.2, on représente la classe *prise-de-tension* par le terme :
$prise\text{-}de\text{-}tension(x_1, x_2 : analyse\ (x_3 : C_0)) \equiv prise\text{-}de\text{-}tension\ (x_1, x_2, x_3)$,
où x_1, x_2, x_3 sont des variables.

Certaines abstractions vont travailler sur des ensembles d'instances d'une même classe. Nous avons choisi de représenter un tel ensemble par la classe d'instanciation des instances entre accolades.

Exemple :
Toutes les instances de la classe *prise-de-tension* sont représentées par :
$\{prise\text{-}de\text{-}tension\ (x_1, x_2, x_3)\}$

Si l'on désire ne représenter qu'un sous-ensemble des instances d'une même classe, il est possible d'instancier les valeurs de certains attributs. Par exemple, l'ensemble des prises de tension dont le premier chiffre est 15 est représenté par : $\{prise\text{-}de\text{-}tension\ (15,x_2, x_3)\}$. L'instance I1 n'appartient pas à cet ensemble.

4.2.2 Représentation des instances dans le cas de l'héritage multiple

Dans le cadre de l'héritage multiple, il existe plusieurs chemins allant d'une instance I à la racine C_0 de la hiérarchie. Le formalisme de représentation d'une instance, dans ce cas, doit faire apparaître ces différents chemins.

Définition 2 : *Soit I une instance multiple, dont les classes d'instanciation sont les classes* $C_{n1}, ..., C_{ns}$. *Elle peut être dénotée par le terme :*

$$C_{n1}\Big(v^1_{n1}, ..., v^{p_{n1}}_{n1} : C_{(n1-1)1}(... : C_0) \cdot ... \cdot C_{(n1-1)r}(... : C_0)\Big) \cdot$$
$$C_{n2}(...) \cdot ... \cdot$$
$$C_{ns}\Big(v^1_{ns}, ..., v^{p_{ns}}_{ns} : C_{(ns-1)1}(... : C_0) \cdot ... \cdot C_{(ns-1)t}(... : C_0)\Big)$$

noté en abrégé

$$C_{n1}(v^1_{n1}, ..., v^{p_{n1}}_{n1}, v^1_{(n1-1)1}, ..., v^{p_{(n1-1)1}}_{(n1-1)1}, ..., v^1_{(n1-1)r}, ..., v^{p_{(n1-1)r}}_{(n1-1)r}, ...) \cdot$$
$$C_{n2}(...) \cdot ... \cdot$$
$$C_{ns}(v^1_{ns}, ..., v^{p_{ns}}_{ns}, v^1_{(ns-1)1}, ..., v^{p_{(ns-1)1}}_{(ns-1)1}, ..., v^1_{(ns-1)t}, ..., v^{p_{(ns-1)t}}_{(ns-1)t}, ...)$$

où le terme $I_j = C_{nj}\Big(v^1_{nj}, ..., v^{p_{nj}}_{nj} : C_{(nj-1)1}(... : C_0) \cdot ... \cdot C_{(nj-1)q}(... : C_0)\Big)$
noté en abrégé $C_{nj}(v^1_{nj}, ..., v^{p_{nj}}_{nj}, ...)$ *modélise l'instance I comme une instance de la classe* C_{nj}.

Exemple :

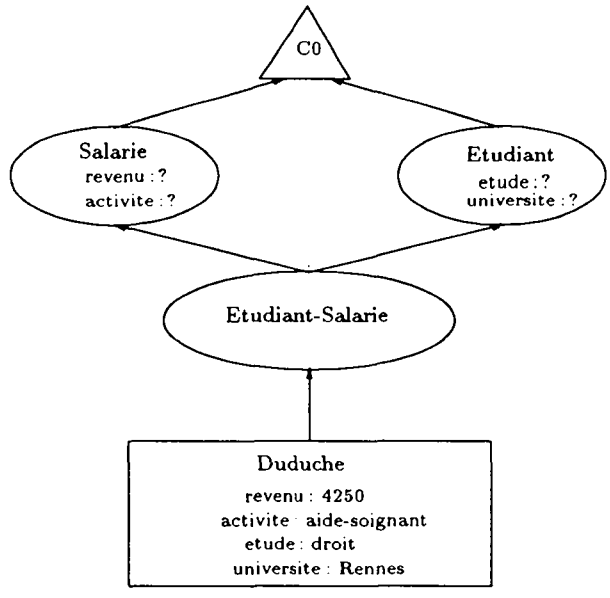


Figure 4.3 : *héritage multiple*

La classe *Etudiant-Salarié* hérite des classes *Salarié* et *Etudiant.*
L'instance *Duduche* est représentée par :
Etudiant-salarié(*Salarié*(4250,aide-soignant : C_0) · *Etudiant*(droit,Rennes : C_0))
noté en abrégé *Etudiant-salarié*(4250,aide-soignant,droit.Rennes)

4.3 Schéma général d'une abstraction

Notre but est de modéliser les diverses formes d'abstraction élémentaires rencontrées au sein d'une R.C.O. Compte-tenu de la représentation choisie pour une instance, la projection d'attributs et la généralisation de classes, qui traduisent un oubli d'information, peuvent se modéliser par une transformation purement **syntaxique** de type règle de réécriture. En revanche, pour la généralisation par élément significatif, qui réduit un ensemble à un élément significatif de cet ensemble, il faut préciser comment se fait le choix de cet élément. De même, pour les abstractions par synthèse, il faut exprimer comment s'effectue la synthèse d'information. Ceci oblige, tout comme le fait Giordana, à introduire un contexte **sémantique**, qui décrit les fonctions calculant les attributs des instances abstraites à partir des attributs des instances de départ.

D'où la définition suivante:

Définition 3: *Une abstraction élémentaire, au sein d'une R.C.O. est une règle de la forme:*
contexte: instance(s)-avant-abstraction → instance(s)-après-abstraction
où **instance(s)-avant-abstraction** *décrit l'instance ou l'ensemble d'instances initial,*
instance(s)-après-abstraction *est soit une instance ou un ensemble d'instances, soit une valeur littérale,*
contexte *décrit les relations entre les attributs des instances abstraites et ceux des instances initiales. Il peut être éventuellement vide. Dans un contexte $\bar{x}$ représente les attributs de l'instance à abstraire et $\bar{y}$ les attributs de l'instance abstraite.*

4.4 L'abstraction dans le cas de l'héritage simple

En utilisant la représentation des instances dans le cadre de l'héritage simple (cf 4.2.1), et la définition générale de l'abstraction dans le cadre des R.C.O. (cf 4.3), nous allons, maintenant, proposer une modélisation pour les différentes formes d'abstractions élémentaires. Nous détaillons, tour à tour, les abstractions par généralisation et les abstractions par synthèse.

4.4.1 Les abstractions par généralisation (g-abstractions)

À une hiérarchie de classes H donnée est associé un ensemble F de règles d'abstraction élémentaires, la *projection d'attributs* et la *généralisation de classes* sont modélisées par des règles génériques implicites. Par contre, la *généralisation par élément significatif* correspond à un ensemble de règles explicites, spécifiques à la hiérarchie H.

L'ensemble des règles de g-abstraction élémentaire associé à H est composé des deux règles implicites de la *projection d'attributs* et de la *généralisation de classes* et de toutes les règles explicites de *généralisation par élément significatif* propres à H.

i) La projection d'attributs

Cette g-abstraction permet d'oublier un attribut d'une instance pour ne conserver que ceux porteurs d'information dans un contexte donné. Elle se modélise par la règle générique suivante:

$$C_n(x_1, ..., x_i, ..., x_p) \xrightarrow[gen\text{-}p/i]{} C_n(x_1, ..., x_{i-1}, x_{i+1}, ..., x_p)$$

qui correspond à un oubli de l'attribut A_i, $i \epsilon [1, p]$. Dans le cas de cette première règle, le contexte est vide. La généricité de cette règle est double. En effet, cette règle peut s'appliquer à une instance quelque soit sa classe d'instanciation C_n et pour une instance donnée, elle peut s'appliquer sur chacun de ses attributs.

Cette règle de réécriture s'applique sur une instance. On lui associe une règle duale s'appliquant sur un ensemble d'instances.

$$\{C_n(x_1, ..., x_i, ..., x_p)\} \quad \underset{gen\text{-}p/i}{\longrightarrow} \quad \{C_n(x_1, ..., x_{i-1}, x_{i+1}, ..., x_p)\}$$

Cette règle signifie que toutes les instances $C_n(x_1, ..., x_i, ..., x_p)$ sont abstraites par la règle *gen-p/i.*

Note: si $p = 1$ alors l'instance I à abstraire est de la forme $C_n(x_1)$ et la règle *gen-p/1* permet alors d'abstraire I par sa classe d'instanciation.

$$C_n(x_1) \quad \underset{gen\text{-}p/1}{\longrightarrow} \quad C_n$$

Exemple:
Si on désire connaître la fréquence des crises d'un patient (cf fig A.7), la seule information utile concernant une instance de la classe *crise* est la *date*. Pour abstraire les instances de *crise*, on utilise alors la règle:

$$\{crise(x_1, x_2)\} \quad \underset{gen\text{-}p/1}{\longrightarrow} \quad \{crise(x_2)\}$$

ii) la généralisation de classes

Cette seconde forme de g-abstraction permet de généraliser une instance d'une classe donnée en une instance de la classe mère de sa classe d'instanciation. La généralisation de classes se traduit alors par la règle de réécriture suivante:

$$C_n(x_n^1, ..., x_n^{P_n}, ..., x_i^1, ..., x_1^{p_1}) \quad \underset{gen\text{-}c}{\longrightarrow} \quad C_{n-1}(x_{n-1}^1, ..., x_{n-1}^{P_{n-1}}, ..., x_i^1, ..., x_1^{p_1})$$

Là encore, dans le cas de cette règle, le contexte est vide. Cette règle est générique, elle peut être appliquée à une instance quelle que soit sa classe d'instanciation.

On associe également à la généralisation de classes une règle duale s'appliquant sur un ensemble d'instances.

$$\{C_n(x_n^1, ..., x_n^{P_n}, ..., x_i^1, ..., x_1^{p_1})\} \quad \underset{gen\text{-}c}{\longrightarrow} \quad \{C_{n-1}(x_{n-1}^1, ..., x_{n-1}^{P_{n-1}}, ..., x_i^1, ..., x_1^{p_1})\}$$

Cette règle réalise une généralisation de classes sur toutes les instances $C_n(x_n^1, ..., x_n^{P_n}, ..., x_i^1, ..., x_1^{p_1})$.

Note: Pour abstraire une instance par l'un de ses ancêtres de rang n, il suffit d'appliquer n fois la règle $gen - c$.

Exemple:

Si on veut connaître toutes les analyses, afin de les utiliser pour effectuer un bilan ou des statistiques (cf fig A.7), il faut oublier les attributs propres aux classes *prise-de-tension* et *prise-de-sang*. Les abstractions utilisées sont alors:

$$\{prise\text{-}de\text{-}tension(x_1, x_2, x_3)\} \xrightarrow[gen\text{-}c]{} \{analyse(x_3)\}$$

et

$$\{prise\text{-}de\text{-}sang(x_1, x_2)\} \xrightarrow[gen\text{-}c]{} \{analyse(x_2)\}$$

iii) La généralisation par élément significatif

Cette g-abstraction s'applique à un ensemble d'instances et permet d'isoler une instance qui possède une propriété, qui la distingue des autres instances de l'ensemble considéré. Tout comme chez Giordana, cette abstraction utilise une fonction f, qui permet de déterminer l'instance à distinguer. La fonction f doit s'appliquer à l'ensemble des ensembles d'attributs de chaque instance de l'ensemble d'instances considéré. Cette abstraction se modélise alors par la règle:

$$\bar{y} = f(\{\bar{x}\}) \ : \ \{C_n(\bar{x})\} \xrightarrow[gen\text{-}e]{} C_n(\bar{y})$$

Exemple:

Si on souhaite abstraire l'ensemble des prises de tension d'un patient en ne retenant que la prise de tension la plus récente, on utilise l'application *plus-récent* qui isole parmi l'ensemble des instances de *prise-de-tension* la plus récente en ne raisonnant que sur les dates de celles-ci. L'abstraction utilisée est alors

$$\bar{y} = plus\text{-}recent(\{\bar{x}\}) \ : \ \{prise\text{-}de\text{-}tension(\bar{x})\} \xrightarrow[gen\text{-}e]{} prise\text{-}de\text{-}tension(\bar{y})$$

4.4.2 Les abstractions par synthèse (s-abstractions)

L'ensemble des règles de s-abstraction élémentaires associé à une hiérarchie de classes sont spécifiques à cette hiérarchie, elles sont donc explicites.

i) La synthèse simple

Le rôle de cette abstraction est de synthétiser une instance d'une classe donnée en une valeur littérale ou en une instance d'une autre classe. L'information abstraite est issue d'une synthèse de l'information initiale. Comme dans la formalisation proposée par Giordana, la valeur littérale ou chaque valeur d'attribut de l'instance créée est le résultat d'une application sur les valeurs d'attributs de l'instance de départ. Cette abstraction se traduit, dans le cas de la synthèse par une instance, par une règle de la forme suivante:

$$y_1 = f_1(\bar{x}), ..., y_q = f_q(\bar{x}) \ : \ C_n(\bar{x}) \quad \underset{\textit{synt-s}}{\longrightarrow} \quad C_m(\bar{y})$$

et, dans le cas de la synthèse par une valeur littérale, par une règle de la forme

$$vl = f_1(\bar{x}) \ : \ C_n(\bar{x}) \quad \underset{\textit{synt-s}}{\longrightarrow} \quad vl$$

A ces deux formes de règles, correspondent deux formes de règles duales, qui s'appliquent à un ensemble d'instances.

$$y_1 = f_1(\bar{x}), ..., y_q = f_q(\bar{x}) \ : \ \{C_n(\bar{x})\} \quad \underset{\textit{synt-s}}{\longrightarrow} \quad \{C_m(\bar{y})\}$$

$$vl = f_1(\bar{x}) \ : \ \{C_n(\bar{x})\} \quad \underset{\textit{synt-s}}{\longrightarrow} \quad \{vl\}$$

Exemple:

On veut abstraire une *prise-de-tension* par un bilan du type "normal", "hypotension" ou "hypertension".

Soit f une application telle que:

$$\begin{aligned} \textit{type-1}^{er}\textit{-chiffre} \quad &\longrightarrow \quad \textit{type-bilan} \\ 14 \leq x \quad &\mapsto \quad \textit{hypertension} \\ 10 \leq x \quad &\mapsto \quad \textit{hypotension} \\ 10 < x < 14 \quad &\mapsto \quad \textit{normal} \end{aligned}$$

L'abstraction peut alors être modéliser par l'une des deux règles suivantes :

$$y_1 = f(x_1) \ : \ \textit{prise-de-tension}(x_1, x_2, x_3) \underset{\textit{synt-s}}{\longrightarrow} \textit{bilan-prise-de-tension}(y_1)$$

$$vl = f(x_1) \ : \ \textit{prise-de-tension}(x_1, x_2, x_3) \underset{\textit{synt-s}}{\longrightarrow} vl$$

La première de ces règles permet d'abstraire une *prise-de-tension* par un instance d'une classe "abstraite" *bilan-prise-de-tension* (cf fig A.7) ; la seconde par une valeur littérale.

ii) La synthèse globale

À partir d'un ensemble d'instances d'une classe donnée, cette s-abstraction crée une instance d'une autre classe ou une valeur littérale. L'information abstraite synthétise les informations de l'ensemble considéré. La valeur littérale, comme chaque valeur d'attribut de l'instance créée est le résultat d'une application sur les ensembles des valeurs des attributs des instances de départ. Cette abstraction se traduit, dans le cas d'une synthèse en une instance, par la règle suivante :

$$y_1 = f_1(\{\bar{x}\}), ..., y_q = f_q(\{\bar{x}\}) \ : \ \{C_n(\bar{x})\} \underset{\textit{synt-g}}{\longrightarrow} C_m(\bar{y})$$

et dans le cas d'une synthèse en une valeur littérale par :

$$vl = f(\{\bar{x}\}) \ : \ \{C_n(\bar{x})\} \underset{\textit{synt-g}}{\longrightarrow} vl$$

Exemples :

• On veut abstraire l'ensemble des crises d'un patient par un bilan général, qui comptabilise le nombre de crises de chaque type (cf fig A.7). En imaginant qu'il en existe de trois types a, b et c, nous avons donc besoin de trois applications, notées par exemple $f_{nb-a}, f_{nb-b}, f_{nb-c}$, qui calculent respectivement le nombre total de crises de type a, b et c. L'abstraction utilisée est alors :

$$y_1 = f_{nb-a}(\{x_2\}), y_2 = f_{nb-b}(\{x_2\}), y_3 = f_{nb-c}(\{x_2\}) \ : \ \{\textit{crise}(\bar{x})\} \underset{\textit{synt-g}}{\longrightarrow} \textit{bilan-des-crises}(\bar{y})$$

• Si on veut abstraire l'ensemble des *prise-de-sang* d'un patient par la moyenne du taux de choslestérol, on utilise l'abstraction suivante :

$$\textit{moyenne} = f_{\textit{moyenne}}(\{x_1\}) \ : \ \{\textit{prise-de-sang}(\bar{x})\} \underset{\textit{synt-g}}{\longrightarrow} \textit{moyenne}$$

4.5 L'abstraction dans le cas de l'héritage multiple

Nous venons de présenter la modélisation retenue pour les diverses abstractions élémentaires, mais ceci dans un contexte d'utilisation des R.C.O. simplifié, plus précisement dans le cas de l'héritage simple. Nous allons à présent, étendre ces modélisations pour une R.C.O. quelconque, c'est-à-dire admettant l'héritage multiple.

Dans les paragraphes 4.4.1 et 4.4.2, nous avons modélisé chacune des abstractions répertoriées. Il s'agit, maintenant, d'étudier les modifications à apporter pour adapter cette modélisation au cas de l'héritage multiple. Ces modifications sont liées à la multiplicité des chemins d'héritage et ne concernent donc que les abstractions liées au chemin d'héritage à savoir la *projection d'attributs* et la *généralisation de classes.*

4.5.1 La projection d'attributs

Rappelons que la *projection d'attributs* permet d'oublier un attribut, pour ne conserver que ceux qui sont porteurs d'information dans un contexte donné.
Elle se modélise par la règle générique suivante:

$$C_{n1}(x_{11},...,x_{p1}) \cdot ... \cdot C_{nj}(x_{1j},...,x_{ij},...,x_{pj}) \cdot ... \cdot C_{ns}(x_{1s},...,x_{ps}) \longrightarrow_{gen\text{-}p/i,j}$$
$$C_{n1}(x_{11},...,x_{p1}) \cdot ... \cdot C_{nj}(x_{1j},...,x_{(i-1)j},x_{(i+1)j},...,x_{pj}) \cdot ... \cdot C_{ns}(x_{1s},...,x_{ps})$$

La généricité de cette règle est triple. Elle peut s'appliquer à une instance quelles que soient ses classes d'instanciation et elle peut s'appliquer sur chacun des attributs de chacune de ses classes d'instanciation, c'est-à-dire pour tout $j \in \{1,..,s\}$ et pour tout $i \in \{1j,..,pj\}$.

Comme dans le cadre de l'héritage simple, la *projection d'attributs* possède une règle duale s'appliquant sur les ensembles d'instances.

4.5.2 La généralisation de classes

Une instance, dans le cadre de l'héritage multiple, peut avoir plusieurs classes d'instanciations. Dans ce cas, la *généralisation de classes* précise la classe d'instanciation à généraliser. Cette g-abstraction se traduit par la règle:

$$C_{n1}(...) \cdot ... \cdot C_{nj}(x^1_{nj},...,x^{p_{nj}}_{nj},v^1_{(nj-1)1},...) \cdot ... \cdot C_{ns}(...) \longrightarrow_{gen\text{-}c/j}$$
$$C_{(nj-1)1}(x^1_{(nj-1)1},...) \cdot ... \cdot C_{(nj-1)r}(x^1_{(nj-1)r},...)$$

où j dénote le rang de la classe d'instanciations à généraliser.

Cette règle de réécriture est doublement générique. Elle peut s'appliquer à toute instance de la hiérarchie et pour une instance donnée, elle peut s'appliquer sur chacune de ses classes

d'instanciation, c'est-à-dire pour tout $j \in \{1, .., s\}$.

Cette règle possède une règle duale s'appliquant sur les ensembles d'instances.

4.6 Les abstractions composées

Nous venons d'exhiber cinq règles d'abstraction, qui permettent de modéliser les abstractions élémentaires au sein d'une R.C.O. Il s'agit pour nous maintenant d'étudier les abstractions composées, c'est-à-dire les compositions d'abstractions élémentaires. Nous développons, successivement, les compositions de g-abstractions, de s-abstractions, de g-abstractions et de s-abstractions. Tout d'abord, nous définissons les types d'abstraction applicables à une instance ou à un ensemble d'instances.

4.6.1 Deux types d'abstraction simples

Pour simplifier les écritures, nous nous plaçons dans le cadre de l'héritage simple. Rappelons qu'une instance I de la classe d'instanciation C_n est représentée en abrégé par $C_n(v_n^1, ..., v_n^{p_n}, ..., v_1^1, ..., v_1^{p_1})$, où v_j^i est la valeur du i^{em} attribut propre de la classe C_j. Cette valeur peut-être soit de type simple (entier, chaîne de caractères,...), soit une instance ou un ensemble d'instances d'une classe C_q, dans le cas où l'attribut est de type C_q. Pour une instance I donnée, on peut appliquer une règle d'abstraction élémentaire

1. soit sur le terme t représentant I,
2. soit sur le sous-terme t_i représentant la valeur du i^{em} attribut.

On distingue donc deux types d'abstractions applicables à une instance ou à un ensemble d'instances.

Définition : *On appelle* **abstraction entière simple** *de l'instance I (ou de l'ensemble d'instances $\{I\}$), l'application au terme t représentant I (ou au terme $\{t\}$ représentant $\{I\}$) de l'une des règles d'abstraction élémentaire.*

Définition : *On appelle* **abstraction partielle simple** *de l'instance I, l'application au sous-terme t_i représentant la valeur v_i de l'attribut A_i de l'une des règles d'abstraction élémentaire.*

4.6.2 Les g-abstractions composées

Nous nous intéressons, tout d'abord, aux compositions de g-abstractions élémentaires.

Définition : *On appelle* **g-abstraction entière** *de l'instance I (ou de l'ensemble d'instances $\{I\}$), toute composition de g-abstractions entières simples de I (ou de $\{I\}$).*

Rappelons que les g-abstractions sont des abtractions sans changement de concept (cf 3.3), et qu'aucune règle de g-abstraction ne transforme une instance (ou un ensemble d'instances), en une instance dépendant d'un tout autre chemin d'héritage.

Ainsi si l'on abstrait une instance I par une g-abstraction composite simple portant sur la valeur I_i de type C_{ni}, alors la valeur I'_i de l'attribut A_i de l'instance abstraite obtenue est également de type C_{ni}. Nous illustrons ceci, par les exemples suivants.

Exemples:

Soient les instances

$I_1 \equiv C_n(v_1, C_m(u_1 : C_{m-1}(u_2 : C_o)), : C_o)$,
notée en abrégé $C_n(v_1, C_m(u_1, u_2))$,
où l'attribut A_2 est de type C_m et a pour valeur $C_m(u_1, u_2)$.

$I_2 \equiv C_n(v_1, \{C_{n1}(u_{11}, u_{12}), C_{n1}(u_{21}, u_{22})\})$,
où l'attribut A_2 est de type C_{n1} multivalué et a pour valeurs $C_{n1}(u_{11}, u_{12})$ et $C_{n1}(u_{21}, u_{22})$.

Si on leur applique les règles suivantes:

$$\bullet\ I_1 \underset{\textit{gen-p/1}}{\longrightarrow} C_n(v_1, C_m(u_2))$$

la valeur $C_m(u_2)$ de l'attribut A_2 est de type C_m.

$$\bullet\ I_1 \underset{\textit{gen-c}}{\longrightarrow} C_n(v_1, C_{m-1}(u_2))$$

la valeur $C_{m-1}(u_2)$ est de type C_{m-1}, l'attribut A_2 est de type C_m, mais C_m étant une sous-classe de C_{m-1}, A_2 est également de type C_{m-1}.

$$\bullet\ \textit{contexte}_n : I_2 \underset{\textit{gen-e}}{\longrightarrow} C_n(v_1, C_{n1}(u_{11}, u_{12}))$$

la valeur $C_{n1}(u_{11}, u_{12})$ est de type C_{n1}.

Définition: *On appelle* **g-abstraction partielle** *de l'instance I toute composition de g-abstractions partielles simples, portant sur un même attribut de I.*

Définition: *On appelle* **g-abstraction composée** *de l'instance I (ou de l'ensemble d'instances $\{I\}$), toute composition de g-abstractions entières et de g-abstractions partielle de I (ou de $\{I\}$).*

4.6.3 Les s-abstractions composées

Nous nous intéressons, à présent, aux compositions de s-abstractions élémentaires.

Définition : *On appelle* **s-abstraction entière** *de l'instance I (ou de l'ensemble d'instances $\{I\}$), toute compositions de s-abstractions entières simples de I (ou de $\{I\}$).*

Contrairement aux g-abstractions, les s-abstractions sont des abstractions avec changement de concept (cf 3.1). Ainsi, une règle de s-abstraction appliquée à une instance I, produit une instance I' dépendant d'un chemin d'héritage totalement différent de celui de I. Par conséquent, si l'on abstrait une instance I par une s-abstraction composite portant sur la valeur I_i de l'attribut A_i de type C_{ni}, le type C_{mi} de l'instance I'_i obtenue ne respecte pas le type C_{ni} de l'attribut A_i. Nous illustrons ceci dans l'exemple qui suit.

Exemple :
En appliquant la règle

$$contexte_1 : C_{n1}(\bar{x}) \underset{synt\text{-}s}{\longrightarrow} C_{m1}(\bar{y})$$

à l'instance $I \equiv C_n(v_1, C_{n1}(u_1, u_2))$, où l'attribut a_2 de type C_{n1} et de valeur $C_{n1}(u_1, u_2)$.

$$I \underset{synt\text{-}s}{\longrightarrow} C_n(v_1, C_{m1}(w_1, w_2))$$

la valeur $C_m(w_1, w_2)$ est de type C_{m1} qui est différent de C_{n1}.

Il est donc impossible de faire une s-abstraction partielle. Par conséquent, les seules s-abstractions composées possibles sont les s-abstractions entières.

4.6.4 Les g/s-abstractions composées

Nous nous intéressons, maintenant, aux abstractions composées issues de compositions de g-abstractions et de s-abstractions.

De par la restriction portant sur les compositions des s-abstractions et de par la définition d'une g-abstraction composée, nous pouvons donner la définition suivante :

Définition : Une abstraction composée *est une composition de g-abstractions composées et de s-abstractions entières.*

Chapitre 5

Conclusion

5.1 Les acquis

Dans le chapitre 3, nous avons exhibé 5 formes d'abstraction élémentaires, au sein d'une R.C.O., regroupées en deux classes :

- **les abtractions par généralisation (g-abstractions)** qui correspondent à un oubli d'information et sont des *abstractions sans changement de concept*, ce sont :
  - la *projection d'attributs*, qui permet de ne garder que les attributs porteurs d'information dans un contexte donné,
  - la *généralisation de classe*, qui permet d'oublier la classe d'instanciation d'une instance ainsi que ses attributs propres,
  - la *généralisation par élément significatif*, qui réduit un ensemble d'instances à un élément significatif de cet ensemble.
- **les abstractions par synthèse (s-abstractions)** qui correspondent à un condensé d'information, et sont des *abstractions avec changement de concept*, ce sont :
  - la *synthèse simple*, qui synthétise les informations d'une instance en une instance d'une autre classe ou en une valeur littérale,
  - la *synthèse globale*, qui synthétise les informations d'un ensemble d'instances d'une classe en une instance d'une autre classe ou en une valeur littérale.

Puis, nous avons montré les similitudes qui existent entre ces formes d'abstractions et les travaux réalisés sur l'abstraction dans le cadre plus général de la résolution de problèmes.

Partant de ces similitudes, nous avons proposé, dans le chapitre 4, une définition de l'abstraction plus adaptée aux R.C.O. dans laquelle une abstraction est représentée par une règle

de réécriture associée à un contexte sémantique. Un contexte décrit les relations entre les attributs des instances initiales et ceux des instances abstraites, à l'aide de fonctions.

A chaque hiérarchie de classes est associé un système de règles d'abstraction composé d'une part de deux règles implicites, correspondant à la *projection d'attributs* et à la *généralisation de clasees*, et d'autre part d'un certain nombre de règles explicites, correspondant à la *généralisation par élément significatif*, à la *synthèse simple* et à la *synthèse globale*. Nous avons montré que toutes les compositions d' abstractions élémentaires sont possibles, si l'on exclus les abstractions par synthèse portant sur un attribut d'une instance.

5.2 Les perspectives

• Nous nous sommes limités aux abstractions qui s'appuient sur la hiérarchie de classes en délaissant volontairement celles s'appuyant sur la hiérarchie de structure. En effet, les abstractions développées conservent le chemin d'héritage de l'instance initiale, ceci dans le cas d'une g-abstraction, ou passe à un chemin d'héritage totalement indépendant de celui de l'instance initiale, ceci dans le cas d'une s-abstraction.

Mais il semble intéressant d'étudier plus attentivement, les abstractions s'appuyant sur la hiérarchie de structure d'une instance. Il s'agit, de pouvoir abstraire une instance par un de ses composants (un attribut étant lui-même une instance), en utilisant une règle de la forme :

$$C_n(I_1, v_2, v_3) \longrightarrow I_1$$

De telles abstractions paraissent pouvoir être utilisées dans des problèmes de diagnostics par exemple. Cette forme d'abstraction reste un point à développer.

• D'autre part, nous remarquons que le contexte d'une règle tel que nous l'avons défini, ne permet pas de modéliser tous les raisonnements. Nous présentons, ci-dessous, deux exemples de raisonnements qui ne sont pas modélisables de par la limitation d'expression des contextes.

1. On ne peut pas définir une règle dont l'ensemble des instances initiales est issu de la restriction d'un ensemble par un de ses sous-ensembles.

   On ne peut pas, par exemple, expliciter une règle d'abstraction par synthèse globale qui permet d'abstraire des prises de tension par une valeur littérale en limitant le domaine d'action de cette règle au sous-ensemble des prises de tension dont le 1^{er} chiffre est supérieur à 15.
   Une règle de la forme :

   $$x_1 > 15 \ ; \ vl = f(\bar{x}) \ : \ \{prise\text{-}de\text{-}tension(x_1, x_2, x_3)\} \longrightarrow vl$$

permettrait de traduire cet exemple.

2. D'autre part, on n'a pas la possibilité de définir une règle dont le contexte utilise les attributs d'instances autres que l'instance à abstraire.

   Aussi, si on considère par exemple, le cas où l'on souhaite abstraire une *prise-de-tension* par une valeur littérale, telle que "à-conserver" ou "à-refaire", en comparant ses attributs aux moyennes des attributs des instances de *prise-de-tension* antérieures, aucune des règles d'abstractions élémentaire présentées ne permet de modéliser un tel raisonnement en se limitant à un contexte qui ne décrit que les relations entre les attributs des instances initiales et ceux des instances abstraites.
   Une règle de la forme :

   $$\textit{moyenne-tension}(x_1, x_2)\ ;\ vl = \textit{test}(\bar{x}, \bar{y})\ :\ \textit{prise-de-tension}(y_1, y_2, y_3) \longrightarrow vl$$

   permettrait de traduire le raisonnement souhaité.

Afin de modéliser un ensemble plus vaste de raisonnements, il semble intéressant d'une part d'intégrer des prédicats logiques dans un contexte pour exprimer une restriction sur l'ensemble des instances initiales à abstraire, et d'autre part qu'un contexte puisse référencer des instances n'apparaîssant pas dans la règle de réécriture. Ces deux points restent également à développer.

Références

[1] G.MASSINI, A.NAPOLI, D.COLNET, D.LEONARD , and K.TOMBRE. *Les langages à objets.* InterEditions, 1989.

[2] Josh D.TENENBERG. Preserving cosistency across abstraction mappings. *IJCAI-87*, 1011–1014, 1987.

[3] R.FIKES and T.KEHLER. The role of frame-based representation in reasoning. *ACM*, 1985.

[4] A.GIORDANA, G.PERETTO, D.ROVERSO , and L.SAITTA. Abstraction : an alternative view of concept acquisition. *Elsevier Science Publishing*, 379–387, 1990.

[5] F.GIUNCHIGLIA and T.WALSH. Abstract theorem proving. *IJCAI-89*, 372–377, 1989.

[6] D.PLAISTED. Theorem proving with abstraction. *Artificial Intelligence*, 16:47–108, 1981.

[7] A.KNOBLOCK. A theory of abstraction for hierarchical planning. In *Change of Representation and Inductive Bias*, Kluwer Publ.Co, 1989.

[8] S.ROBIN and M.COURANT. Mécanismes d'abstraction dans une représentation de connaissances objectale. 1992. rapport interne.

[9] R.VOR'CH . Généralisation et abstraction : une étude en calcul propositionnel. In *Première rencontre nationales des jeunes chercheurs en Intelligence Artificielle*, pages 182–195, IRISA, September 1992.

[10] R.NOUSSI. *un modèle objectal pour la synthèse de dossiers médicaux.* PhD thesis, Université de Rennes1, 1989.

Annexe A

Figures

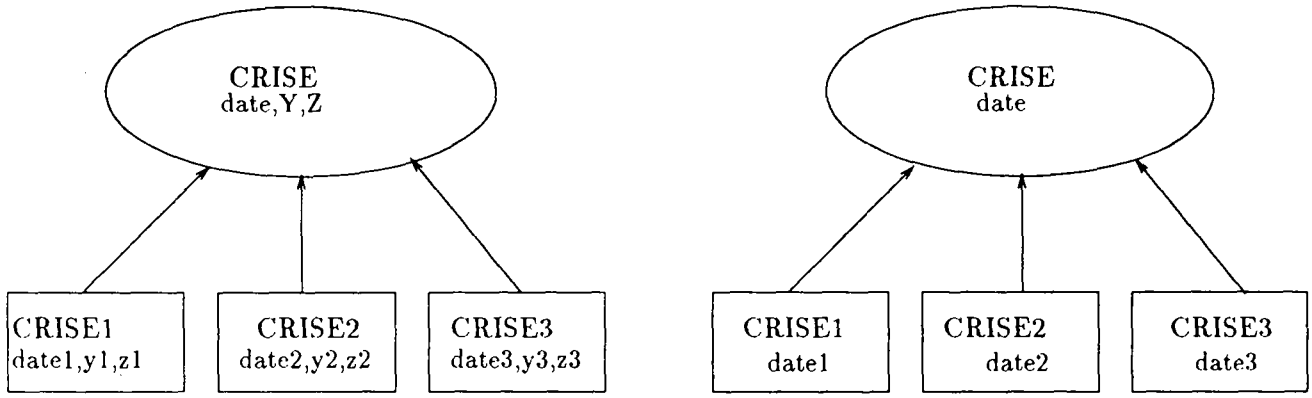


fig 1a

fig 1b

Figure A.1 : *projection d'attributs*

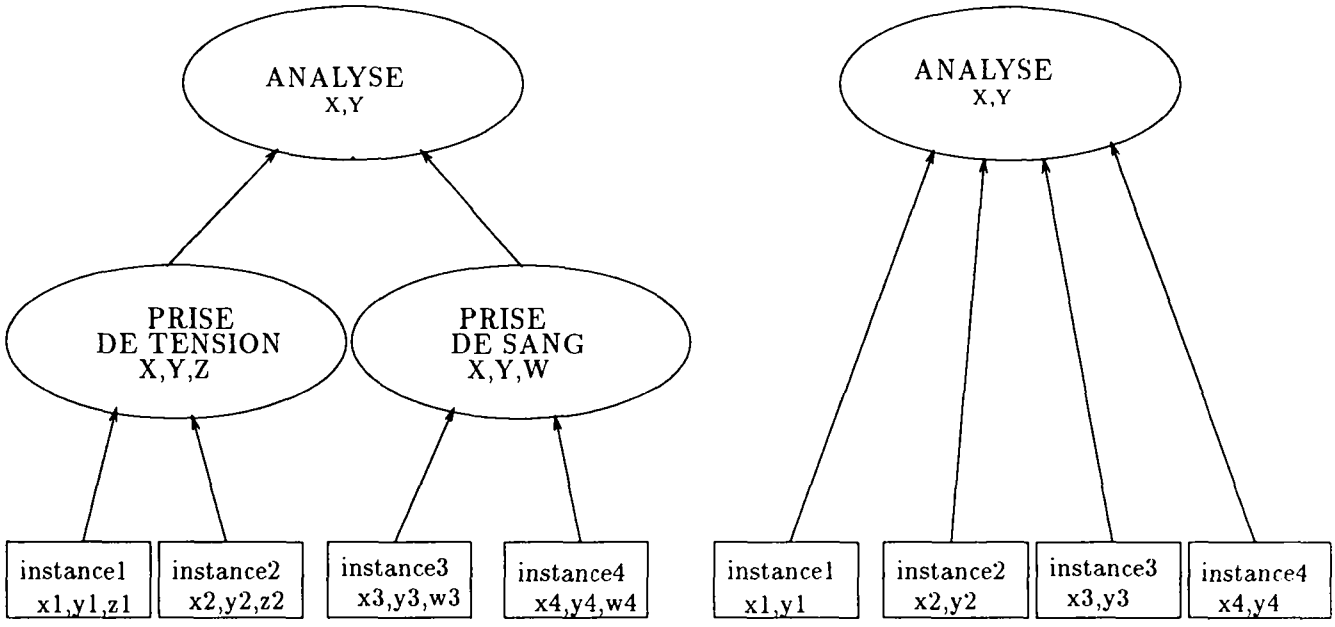


fig 2a

fig 2b

Figure A.2 : *généralisation de classes*

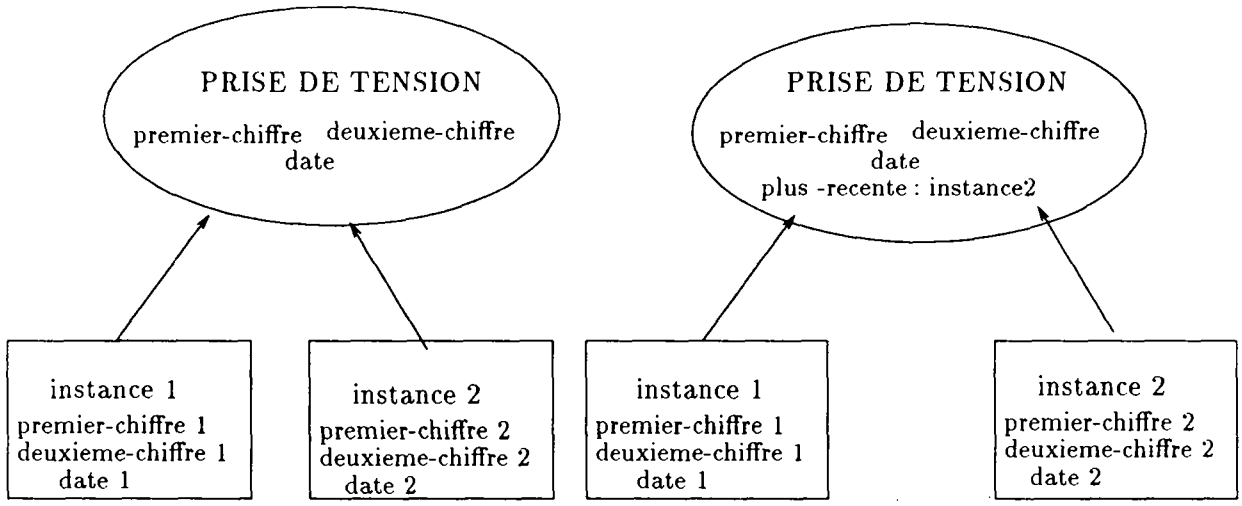


fig 3a fig 3b

Figure A.3 : *généralisation par un élément significatif*

commentaires de la figure A.3 :

Dans le cas, où l'on veut conserver le résultat d'une telle abstraction, il paraît difficile de créer une nouvelle classe qui regroupe les instances élues, car aucune instance n'appartiendrait définitivement à cette sous-classe. Pour cet exemple, l'arrivée d'une nouvelle instance de *prise-de-tension* peut détrôner, celle qui était jusque-là la plus récente.
Il nous paraît plus raisonnable d'ajouter un attribut propre à la classe mère qui n'est pas hérité par les instances, dont la valeur est l'instance distinguée. Cet attribut, s'il est conservé devra être remis à jour dès la venue d'une nouvelle instance.
Note: L'utilisation d'un attribut particulier adapté à ce genre de fonctionnalité a été décrit par Fikes sous le terme de ownslot [3].

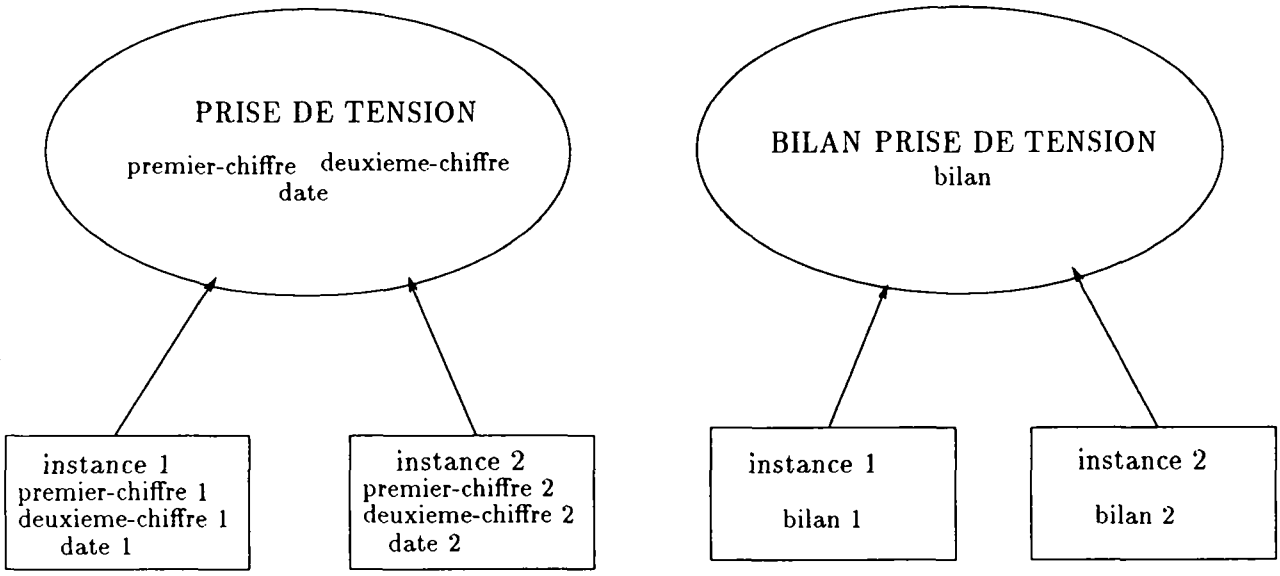
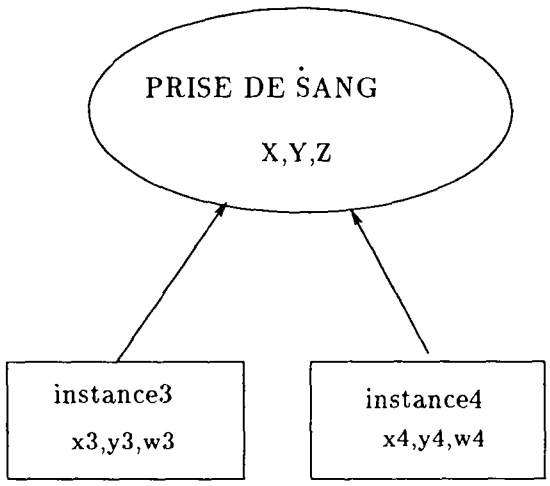


fig 4a

fig 4b

Figure A.4 : *synthèse simple*



MOYENNE
(entier)

fig 5a

fig 5b

Figure A.5 : *synthèse globale*

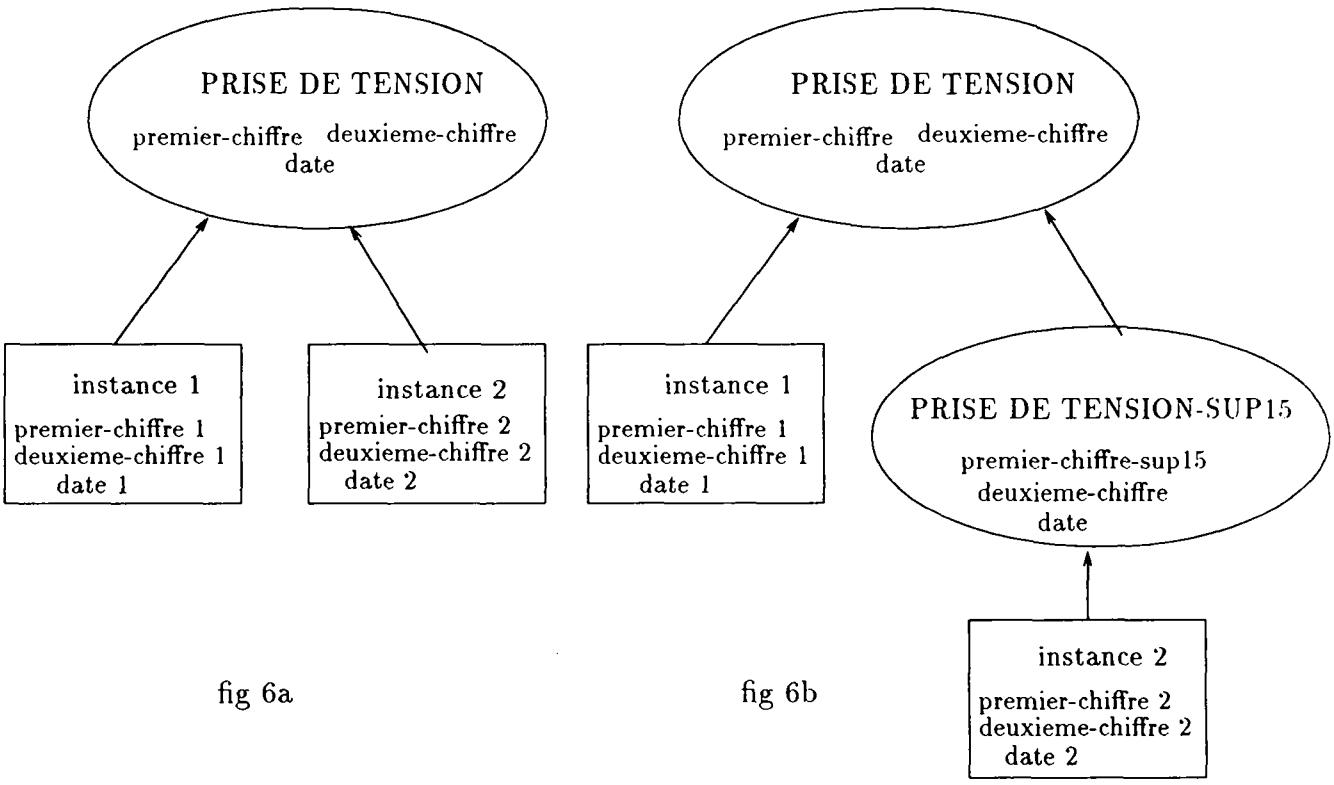


Figure A.6 : *détermination de sous-ensemble*

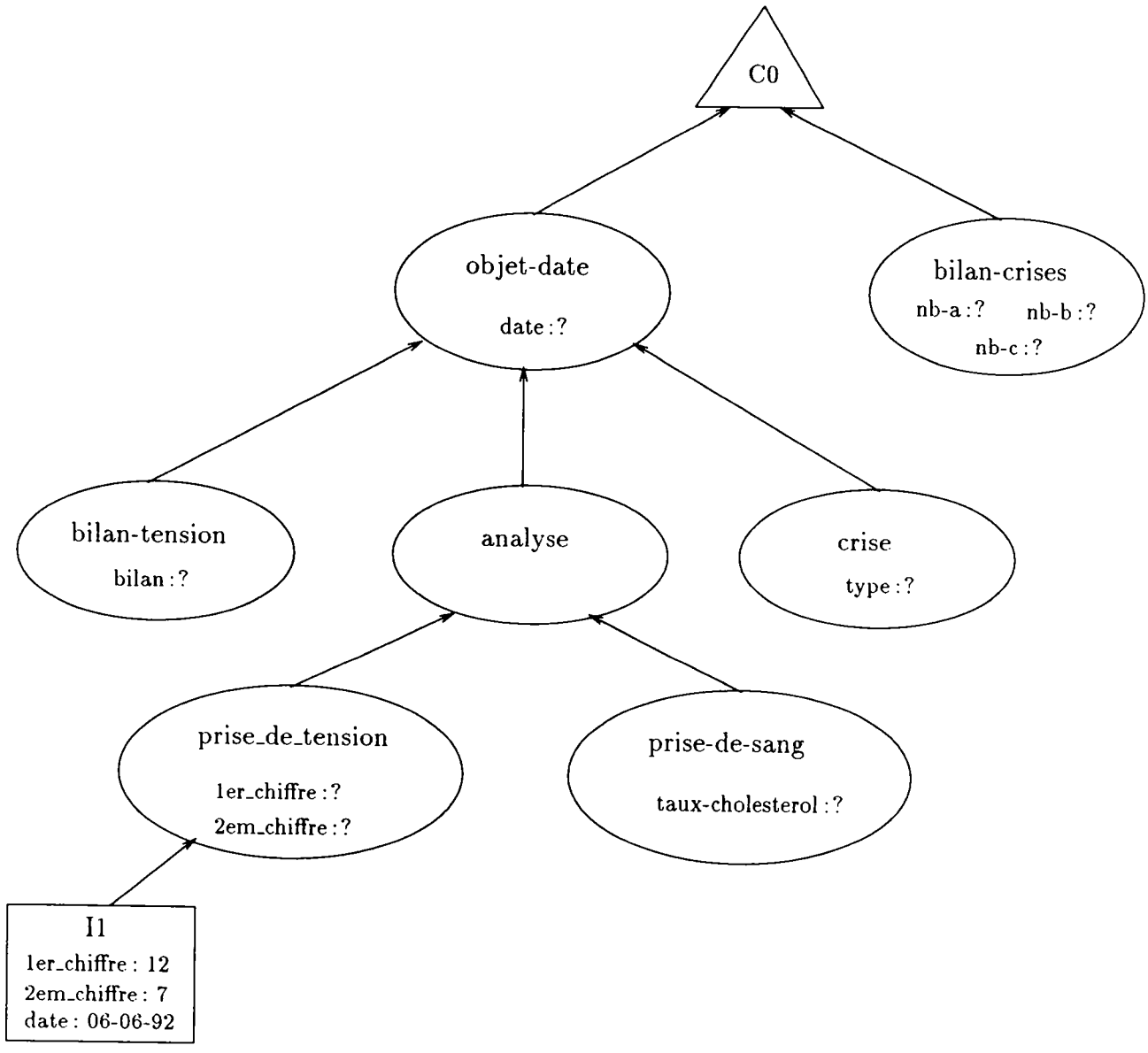


Figure A.7 : *base de connaissances*

LISTE DES DERNIERES PUBLICATIONS INTERNES IRISA PARUES EN 1992

PI 687 THE COMPILATION OF PROLOG and its Execution with MALI
Pascal BRISSET, Olivier RIDOUX
Novembre 1992, 90 pages.

PI 688 GENERALISATION DE L'ANALYSE FACTORIELLE MULTIPLE A L'ETUDE DES TABLEAUX DE FREQUENCE ET COMPARAISON AVEC L'ANALYSE CANONIQUE DES CORRESPONDANCES
Lila ABDESSEMED, Brigitte ESCOFIER
Novembre 1992, 34 pages.

PI 689 OVERVIEW OF THE KOAN PROGRAMMING ENVIRONMENT FOR THE iPSC/2 AND PERFORMANCE EVALUATION OF THE BECAUSE TEST PROGRAM 2.5.1..
François BODIN, Thierry PRIOL
Décembre 1992, 16 pages.

PI 690 CONCURRENT PROGRAMMING NOTATIONS IN THE OBJECT-ORIENTED LANGUAGE ARCHE
Marc BENVENISTE, Valérie ISSARNY
Décembre 1992, 34 pages.

PI 691 COMMUNICATION EFFICIENT DISTRIBUTED SHARED MEMORIES
Masaaki MIZUNO, Michel RAYNAL, Gurdip SINGH, Mitchell L. NEILSEN
Décembre 1992, 24 pages.

PI 692 ETUDE COMPAREE DES ARCHITECTURES DES MICROPROCESSEURS MIPS R4000, DEC 21064 ET T.I. SUPERSPARC
André SEZNEC, Anne-Marie KERMARREC, Thierry VAULEON
Décembre 1992, 106 pages.

PI 693 ETUDE DE L'UTILISATION D'ARCHITECTURES PARALLELES A MEMOIRE DISTRIBUEE POUR LA REALISATION DE COMMUTATEURS DE RESEAUX HAUT DEBIT
Jean-Marc JEZEQUEL, Claude JARD, Thierry ESTIMBRE
Décembre 1992, 40 pages.

Unité de Recherche INRIA Rennes
IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399